Improved Parallel Mask Refreshing Algorithms
Generic Solutions with Parametrized Non-Interference & Automated Optimizations

Gilles Barthe[1], Sonia Belaïd[2], François Dupressoir[3], Pierre-Alain Fouque[4], Benjamin Grégoire[5], François-Xavier Standaert[6], and Pierre-Yves Strub[7]

[1] IMDEA Software Institute, Spain
[2] CryptoExperts, France
[3] University of Surrey, United Kingdom
[4] Université de Rennes 1, France
[5] Inria Sophia-Antipolis – Méditerranée, France
[6] Université catholique de Louvain, ICTEAM Institute, Belgium
[7] Ecole Polytechnique, France.

Abstract. Refreshing algorithms are a critical ingredient for secure masking. They are instrumental in enabling sound composability properties for complex circuits, and their randomness requirements dominate the performance overheads in (very) high-order masking. In this paper, we improve a proposal of mask refreshing algorithms from EUROCRYPT 2017, that has excellent implementation properties in software and hardware, in two main directions. First, we provide a generic proof that this algorithm is secure at arbitrary orders – a problem that was left open so far. We introduce Parametrized Non-Interference as a new technical ingredient for this purpose, that may be of independent interest. Second, we use automated tools to further explore the design space of such algorithms and provide the best known parallel mask refreshing gadgets for concretely relevant security orders. Incidentally, we also prove the security of a recent proposal of mask refreshing with improved resistance against horizontal attacks from CHES 2017.

Keywords: Side-channel attacks, masking countermeasure, refreshing algorithms, composability.

1 Introduction

State-of-the-art. Side-channel attacks are an important threat to the security of cryptographic hardware [22]. While originally mostly applied to small embedded devices, they recently targeted an increasingly large spectrum of implementations, such as general-purpose computers or smartphones [26,19,1,16]. So the understanding of these attacks and solutions to prevent them is an important challenge for modern security applications. In this context, masking is now established as a widely used countermeasure allowing cryptographic designers to mitigate side-channel attacks. Its main idea consists in splitting any sensitive

value into several shares so that the adversary needs to collect and combine information about all the shares to extract secrets from the leakages. One key reason for the success of masking is the good theoretical understanding it allows. In particular, the security of masking is now shown in various security models, ranging from the abstract probing model of Ishai et al. [20], the noisy leakage model of Prouff and Rivain [23], and the bounded moment model of Barthe et al. [5]. Additionally, fundamental steps have been made in the connection of these models. In particular, the work of Duc et al. showed that security in the most abstract (but easiest to manipulate) probing model implies security in the most practically relevant (but more involved) noisy leakage model [12]. One important consequence of this work is that it is now possible to verify the security of masked implementations based on their abstract description, and to translate these abstract guarantees into concrete security [13], by checking two additional conditions: the independence of the shares' leakages (which is typically assessed in the bounded moment model) and a sufficient noise level (which is naturally assessed in the noisy leakage model). The latter results suggest that obtaining masking schemes that are secure in the (abstract) probing model is anyway a good preliminary (if not a strict requirement) for the design of secure masked implementations.

Yet, and despite its appealing conceptual simplicity, reasoning about security in the probing model is challenging, because the number of possible observation sets in a masked implementation incurs an exponential growth when the number of shares increases. This difficulty is typically illustrated by subtle bugs in early works and hand-made proofs on masking [25,24], corrected in [10,11]. As a result, the state-of-the-art proofs of masked implementations now generally combine two types of approaches.

The first one is to rely exclusively on *composable gadgets*, as formalized by Barthe et al. [4]. The latter made a significant step towards sound compositional reasoning, by introducing Strong Non-Interference (SNI). Informally, SNI refines previous notions of probing security, by separating between external and internal observations and by requiring that the number of shares needed to simulate an observation set is upper bounded by the number of internal observations. It provably allows connecting masked gadgets without risks of compositional flaws (at the cost of performance overheads and randomness requirements), and therefore simplifies the analysis of full implementations.

The second one is to carry out more *global compositional proofs*. It generally allows improving the performances and reducing the randomness of masked implementations, this time at the cost of a more challenging analysis. In this case, the implementations typically mix SNI gadgets with (less demanding) NI gadgets (i.e., a relaxation of SNI where the the number of shares needed to simulate an observation set is upper bounded by the total number of observations). Evaluations can then take advantage of formal methods to better deal with the large number of cases to be covered by the proofs [3].

One essential ingredient for both approaches are so-called *refreshing algorithms* which are instrumental in compositional reasoning [7], and generally allow

"splitting" a masked implementation in small (enough) parts that can then be analyzed globally. Recent implementations of (very) high-order masking schemes have shown that the (randomness) cost of these refreshing algorithms now accounts for a significant part of the global performance overheads [17,21]. In this paper, we therefore tackle the generic improvement and automated optimization of such refreshing algorithms.

Contribution. Our starting point is a parallel mask refreshing algorithm which was conjectured to be SNI in [5] and has good implementation features for software (and hardware [18]) implementations. In particular, it can easily be implemented with simple operations such as rotations and XORs that are available on most devices. Our contributions in this respect are threefold.

First, we observe that existing notions of NI and SNI may lack in granularity to analyze the security of this refreshing. We therefore introduce a novel notion of Parametrized Non-Interference (f-NI). Informally, f-NI maintains the distinction from SNI between external and internal observations, but requires that the number of shares needed to simulate an observation set is bounded by a function f of the number of internal observations. The definition of f-NI subsumes SNI (by setting f to be the identity function on naturals smaller than the masking order) and NI (by setting f to be the function that maps all naturals smaller than the masking order to the masking order).

Second, we leverage this new notion of f-NI in order to answer in the positive the conjecture from [5], and show that their (iteration of) parallel mask refreshing gadget(s) is indeed SNI.

Third, we amplify our results on efficient mask refreshing gadgets using synthesis-based techniques to explore the design space of parallel mask refreshing gadgets. The synthesis approach exploits recent works on automatically proving security of masked implementations in the probing model [3,4]. It uses a combination of ideas from program synthesis: intuition-guided templates to selectively reduce the space of gadgets to explore, a scoring system to prioritize the search, a partial order reduction to analyze only once gadgets that differ in inessential details, and efficient data structures to improve the verification of individual gadgets. The overall approach delivers gadgets that outperform the state-of-the-art for (concretely useful) security orders between 5 and 10 (and for larger orders if plugged in the recursive construction of Battistello et al. [6]).

2 Parametrized Non-Interference

Previous work introduced the notion of t-SNI, which supports reasoning compositionally about probing security. It refines t-NI security with a measured independence between observations on inputs and outputs which appears to be just enough to safely compose. Nevertheless, intermediate security notions could be used to provide more flexible and precise compositional reasoning with better performance. For this purpose, we introduce the notion of *parameterised non-interference* (or (t, f^t)-NI), a generalization of both t-SNI and t-NI.

2.1 Definition and Discussion

In the following, we adopt the "non-interference" style of definitions for probing security introduced by Barthe et al. [4], but also further clarify their relationship with more standard definitions of probing security, such as those of Faust et al. [14], which are also composable, or those of Prouff and Rivain [23] and Ishai, Sahai and Wagner [20], which are not.

We first recall some basic definitions on gadgets.

Given a set $\mathbb{K}$ equipped with at least a group structure, we call m-encodings in $\mathbb{K}$ vectors of m elements in $\mathbb{K}$. In practice, such encodings are related to a particular encoding scheme, which we often assume to be additive (that is, an encoding $\mathbf{a}$ is an encoding of some value $a \in \mathbb{K}$ iff $a = \sum_{i=0}^{m-1} \mathbf{a}^i$, where $\sum$ is the iterated addition in $\mathbb{K}$ and $\mathbf{a}^i$ is the ith element in vector $\mathbf{a}$). However, many of the techniques and arguments can be generalised to other encoding schemes. We often work in the scenario where $m = t + 1$ and prefer t-based notation throughout, except where we deviate from this scenario.

A *gadget* G is a probabilistic algorithm that takes as input n m-encodings (where n is the *arity* of the gadget) and returns a single m-encoding, its output. (All our definitions and results can be generalised, at some formal cost, to multi-output gadgets.) We define the semantics of an n-ary gadget G as the function $[\![G]\!]$ that, on input n m-encodings $\mathbf{a_1}, \ldots, \mathbf{a_n}$, returns the joint distribution of *all* intermediate variables (including those used to form the output encoding). In the following, we assume that all intermediate variables are given a distinct *index* in the gadget (for example, the line at which they are defined), and use that index to refer to wires. We call *output positions* those indices that correspond to variables that are used as outputs, and *internal positions* those indices that correspond to wires that do not serve as outputs.

Given an indexed set of values or joint distribution Ω and a set of indices $\mathcal{I}$, we finally denote with $\Omega_{|\mathcal{I}}$ the projection or marginal distribution of those elements of Ω selected by $\mathcal{I}$. For example, the set $\{a, b, c\}$ indexed by (0-indexed) position, could be restricted as follows $\{a, b, c\}_{|\{1,2\}} = \{b, c\}$.

Based on these definitions, we formalize f-NI as follows:

Definition 1 (Parameterised Non-Interference) *A gadget G is (t, f^t)-NI, with $f^t : \mathbb{N}^2 \to \mathbb{N}$, whenever, for any set of probes $\Omega = \widehat{\mathcal{O}} \cup \mathcal{O}$ with $\widehat{\mathcal{O}}$ a set of variables internal to G, and $\mathcal{O}$ a set of output variables of G such that $|\Omega| \leq t$, there exists a set of input indices $\mathcal{I}$ such that:*

1. *$\|\mathcal{I}\| \leq f^t(|\widehat{\mathcal{O}}|, |\mathcal{O}|)$, and*
2. *for any two sets of inputs $(\mathbf{a_i})_{0 \leq i < n}$, $(\mathbf{b_i})_{0 \leq i < n}$ to G, we have*

$$(\mathbf{a_i})_{|\mathcal{I}} = (\mathbf{b_i})_{|\mathcal{I}} \Longrightarrow ([\![G]\!](\mathbf{a_i}))_{|\Omega} = ([\![G]\!](\mathbf{b_i}))_{|\Omega}.$$

Informally, a gadget G is (t, f^t)-NI if, for any observation set d, there exists a subset of input shares $\mathcal{S}$ (whose size is appropriately bounded) that is such that any two inputs that agree on shares in $\mathcal{S}$ will produce the same joint distribution

on variables in d through G. When clear from context, we will often omit the t parameter, simply writing (t, f)-NI, or even f-NI.

We note in particular that this property must be established *without* prior knowledge or assumptions on the distribution of input shares. This makes it distinct (and indeed strictly stronger, as demonstrated in the long version of Barthe et al. [2]) from more local notions such as the tth-order security by Rivain and Prouff [24]. The latter simply requires that any t-tuple of intermediate variables in an implementation is independent of any sensitive variable. For simplicity, we next denote this basic security requirement t-probing security.

We note also that Ishai, Sahai and Wagner, in their proofs, make use of an intermediate notion, that Carlet et al. [8] also call *perfect t-probing security*, and which they define based on one's ability to *simulate* observations based on a subset of an algorithm's input shares. More precisely, they prove on gadgets and algorithms that "any set of $d < t$ probes can be simulated with at most t shares of each input". The meaning of simulation, if taken in the traditional cryptographic sense of an interactive algorithm that should be indistinguishable from the real one, requires specifying which information is made available to the distinguisher. Compositional reasoning requires that the distinguisher can access both the (simulated) probes and input shares. We use a simpler interpretation of the word to denote a (mathematical) function that takes at most t inputs and calculates a joint distribution that is perfectly equal to that produced on its probes by the gadget or algorithm under study, *for any* set of input encodings that coincide on those shares given to the simulator.

Simulation and Non-Interference. When interpreted in this sense, perfect t-probing security is in fact equivalent to (t, f_0^t)-NI, with $f_0^t(t_1, t_2) = \max(t_1 + t_2, t)$. Going further, and using the non-interference-based notions, Belaïd et al. [7] prove that (t, f_0^t)-NI (and thus perfect t-probing security) is in fact equivalent to Barthe et al.'s baseline non-interference notion, t-NI [4]. We note here that f-NI is a strict generalisation of t-NI and t-SNI.

Remark 1 *A gadget is t-NI iff it is (t, f_{NI}^t)-NI with*

$$f_{NI}^t : (t_1, t_2) \mapsto \begin{cases} t_1 + t_2 & \text{if } t_1 + t_2 \leq t \\ \infty & \text{otherwise.} \end{cases}$$

Remark 2 *A gadget is t-SNI iff it is (t, f_{SNI}^t)-NI with*

$$f_{SNI}^t : (t_1, t_2) \mapsto \begin{cases} t_1 & \text{if } t_1 + t_2 \leq t \\ \infty & \text{otherwise.} \end{cases}$$

2.2 First Compositional Results

We now illustrate how one can reason about the security of compositions of f-NI gadgets by stating a simple composition result.

Proposition 1 *Let F_1, F_2 be two single input/single output gadgets. If F_i is (t, f_i)-NI, then the sequential composition $F_1; F_2$ is such that, for any observation set $\Omega = I_1 \uplus I_2 \uplus O$, where I_i contains observations internal to F_i and O contains output observations and such that $|\Omega| \leq t$, the joint distribution of Ω can be predicted using at most $f_1(|I_1|, f_2(|I_2|, |O|))$ shares of the circuit's input.*

Proof. Observations in $I_2 \cup O$ can be simulated using at most $f_2(|I_2|, |O|)$ shares of F_2's input. We now need to simulate observations in I_1, jointly with the $f_2(|I_2|, |O|)$ shares of F_1's output we need in order to simulate F_2. Since F_1 is f_1-NI, this can be done using at most $f_1(|I_1|, f_2(|I_2|, |O|))$ shares of the circuit's inputs. We conclude that observations in $F_1; F_2$ can be simulated using at most $f_1(|I_1|, f_2(|I_2|, |O|))$ shares of its input. □

Proposition 1 is central to some of the compositional proofs given in the rest of this paper. However, it does not clearly show that the composition of two f-NI gadgets is also f-NI for some well chosen f. The following Corollary clarifies this, and justifies our claim that (t, f)-NI is composable. It is a direct consequence of Proposition 1.

Corollary 1 *Let F_1, F_2 be two single input/single output gadgets. If F_i is f_i-NI, then the sequential composition $F_1; F_2$ is $(t, f_{1;2})$-NI, where*

$$f_{1;2}(t_1, t_2) = \max \{f_1(u_1, f_2(u_2, t_2)) \mid u_1 + u_2 = t_1\}.$$

We note that the mere existence of this compositional result does not make f-NI an immediate replacement for NI or SNI: indeed, the composition result shown here exhibits a combinatoric growth in the number of cases to consider when performing a compositional analysis on a circuit (growing with the number of composed gadgets). When composing gadgets that are only NI or SNI, analysis is more efficient but slightly coarser.

2.3 A Closure Property

In addition to composition, (t, f)-NI also enjoys an interesting closure property, similar to that enjoyed by Barthe et al.'s more general notion of $(\mathcal{I}, \mathcal{O})$-NI [4].

Proposition 2 *If a gadget G is both f_1-NI and f_2-NI, then it is also $(f_1 \cap f_2)$-NI with*

$$(f_1 \cap f_2)(t_1, t_2) = \min(f_1(t_1, t_2), f_2(t_1, t_2)).$$

Proof. The result is a direct consequence of the definition for (t, f)-NI. □

Although this property is not used in this paper, we present it here as a general property of (t, f)-NI that may be useful in other applications of the notion.

3 Generic Parallel Mask Refreshing

In this section, we formally analyze, for all t the iteration of regular (parallel) refreshing gadgets proposed and verified at low orders by Barthe et al. [5]. In particular, we prove that the refreshing gadget that successively adds $\lceil t/3 \rceil$ independent encodings of 0 to its encoded input is in fact t-SNI for all t.

Our security proof leverages the notion of f-NI and its compositional properties. Specifically, we first prove that the core additive mask refreshing is f-NI for some appropriate f, and then apply the composition result to conclude that the gadget itself is t-SNI. Beyond its intrinsic interest, this example illustrates the value of the f-NI notion as a generalization of both NI and SNI, but also as a fine-grained property that can be used to compositionally analyse the probing security of gadgets.

3.1 The RefreshBlock Gadget.

Our starting point is the RefreshBlock algorithm shown in Algorithm 1, where indexing is modulo $t+1$, and which corresponds to the building block proposed by Barthe et al. [5] (i.e., Algorithm 1 in this reference). Our description of this algorithm uses slightly different notations, and reorders some of the computation, but we note that our algorithm and Barthe et al.'s [5] compute exactly the same intermediate values and output expressions, and therefore have equivalent probing security (including f-NI and SNI) and functionality. For clarity, we also describe this functionality by giving the expressions for each of the output shares $\mathbf{c}^i$ in terms of the input shares $\mathbf{a}^i$ and fresh random masks $\mathbf{r}^i$:

$$\begin{aligned}
\mathbf{c}^0 &= \mathbf{a}^0 \oplus \mathbf{r}^0 \oplus \mathbf{r}^t,\\
\mathbf{c}^1 &= \mathbf{a}^1 \oplus \mathbf{r}^1 \oplus \mathbf{r}^0,\\
\mathbf{c}^2 &= \mathbf{a}^2 \oplus \mathbf{r}^2 \oplus \mathbf{r}^1,\\
&\vdots\\
\mathbf{c}^t &= \mathbf{a}^t \oplus \mathbf{r}^t \oplus \mathbf{r}^{t-1}.
\end{aligned}$$

Alg. 1 Refresh block algorithm [5].

```
function RefreshBlock(a)
    for i = 0 to t do
        r^i ←$ 𝕂
    for i = 0 to t do
        a^i ← a^i ⊕ r^i
        a^i ← a^i ⊕ r^{i−1}
    return a
```

In addition to its obvious t-NI qualities, we prove that this gadget enjoys a slightly stronger security property, which intuitively constrains an adversary to always place at least two internal probes in order to preserve any information it may have obtained about the gadget's output shares. However, and importantly, even when this condition on the adversary's choice of probes is fulfilled, the adversary will always lose information corresponding to one probe through this gadget.

Proposition 3 *Gadget RefreshBlock is f_{RB}-NI where:*

$$f_{\mathsf{RB}} : (t_1, t_2) \longmapsto \begin{cases} t_1 & \text{if } t_1 \in \{0,1\} \text{ or } t_2 = 0, \\ t_1 + t_2 - 1 & \text{otherwise.} \end{cases}$$

Proof. Gadget RefreshBlock has four kinds of intermediate variables: $\{\mathbf{r}^i\}_{i\leq t}$, $\{\mathbf{a}^i\}_{i\leq t}$, $\{\mathbf{a}^i \oplus \mathbf{r}^i\}_{i\leq t}$ (denoted by $\mathbf{b}^i$), and $\{\mathbf{a}^i \oplus \mathbf{r}^i \oplus \mathbf{r}^{i-1}\}_{i\leq t}$ (denoted by $\mathbf{c}^i$). The first three categories gather internal intermediates variables while the last one refer to output variables.[8]

If $t_2 = 0$ then the observations are only internal variables that contain each at most one input share. Therefore, we can perfectly simulate them using the corresponding input share. We thus need at most t_1 input shares to simulate t_1 internal observations.

Assume now that $t_1 \leq 1$. We denote by *non-input observation* any observation that is not an input share. We start by assigning, to any possible non-input observation $\mathfrak{m}$, a set of indices $\overline{\mathfrak{m}}$:

$$\overline{\mathbf{r}^k} \triangleq \{k\}, \qquad \overline{\mathbf{b}^k} \triangleq \{k\}, \qquad \overline{\mathbf{c}^k} \triangleq \{k, k-1\}.$$

We extend this notion canonically to sets of observations as follows: $\overline{O} \triangleq \bigcup_{\mathfrak{m}\in O} \overline{\mathfrak{m}}$ for any set of non-input observations O. We now prove that if O is a set of non-input observations that contains at most an internal observation, then O can be simulated using no input shares and using randoms in $\{\mathbf{r}^i \mid i \in \overline{O}\}$. If $O = \emptyset$, we are done. Otherwise, assume first that there exists $\mathfrak{m} \in O$ and an index $k \in \overline{\mathfrak{m}}$ s.t. $k \notin \overline{O_{\mathfrak{m}}}$, where $O_{\mathfrak{m}} = O \setminus \{\mathfrak{m}\}$. By induction hypothesis, $O_{\mathfrak{m}}$ can be simulated using no input shares and using randoms in $\mathcal{R} \triangleq \{\mathbf{r}^i \mid i \in \overline{O_{\mathfrak{m}}}\}$. By a direct case analysis on $\mathfrak{m}$, it is clear that $\mathfrak{m}$ can be simulated using the random $\mathbf{r}^k \notin \mathcal{R}$. Hence, we obtain that O can be simulated using no input shares and using randoms in $\{\mathbf{r}^i \mid i \in \overline{O}\}$. We now prove that there always exists such an observation $\mathfrak{m}$. If O is a singleton set, then we take the unique element of O for $\mathfrak{m}$. Otherwise, O contains at least two observations and at least one of them must be an external observation. This, coupled with $|O| \leq t$, implies the existence of an external observation $\mathbf{c}^k \in O$ s.t. $\mathbf{c}^{k+1} \notin O$. If $k \notin \overline{O_{\mathbf{c}^k}}$, we take $\mathbf{c}^k$ for $\mathfrak{m}$. Otherwise, for k to occur in $\overline{O_{\mathbf{c}^k}}$, since $\mathbf{c}^{k+1} \notin O$, it must be that O contains one (and exactly one by assumption on O) internal observation of the form $\mathbf{b}^k$ or $\mathbf{r}^k$. Consider the maximal chain $\mathbf{c}^{k-i}, \ldots, \mathbf{c}^k$ of external observations of O.

[8] In our notations, $\mathbf{x}^i$ always has to be interpreted modulo $t+1$, i.e. $\mathbf{x}^i \triangleq \mathbf{x}^{i \bmod t+1}$.

Since O is of maximum size t, the length $i+1$ of this chain is at most t. Hence, $k-i-1 \neq k$ and $k-i-1$ cannot be in $\overline{\mathbf{b}^k}$ or $\overline{\mathbf{r}^k}$. Hence, for $k-i-1$ to be in $\overline{O_{\mathbf{c}^{k-i}}}$, $\mathbf{c}^{k-i-1}$ must be in O, which is impossible by maximality of the chain. If follows that $k-i-1 \notin \overline{O_{\mathbf{c}^{k-i}}}$, and we can use $\mathbf{c}^{k-i}$ for $\mathfrak{m}$. We now conclude the case $t_1 \leq 1$. If $t_1 = 0$ or $t_1 = 1$ with the internal observation not being an observation of an input share, we conclude by directly applying the fact we just proved. If t_1 and we observe an input share $\mathbf{a}^k$, we first simulate the input share using $\mathbf{a}^k$ and no randoms, and then, applying again the previously proven fact, simulate the outputs using only randoms.

Now, assume that $2 \leq t_1$ and $1 \leq t_2$. To any observation $\mathfrak{m}$, we attach an optional input share dependency $\widehat{\mathfrak{m}}$ s.t. $\mathfrak{m}$ can be perfectly simulated from $\widehat{\mathfrak{m}}$ by simply using the arithmetical expression of $\mathfrak{m}$:

$$\widehat{\mathbf{r}^k} \triangleq \bot, \qquad \widehat{\mathbf{a}^k} \triangleq \widehat{\mathbf{b}^k} \triangleq \widehat{\mathbf{c}^k} \triangleq \mathbf{a}^k.$$

Again, we extend this notion canonically to sets of observations as follows: $\widehat{O} \triangleq \{\widehat{\mathfrak{m}} \mid \mathfrak{m} \in O, \widehat{\mathfrak{m}} \neq \bot\}$ for any set of observations O. It is clear that if $\mathfrak{m}_1$ and $\mathfrak{m}_2$ are two observations that we simulate in isolation using their respective arithmetical expressions and the input-shares $\widehat{\mathfrak{m}_1}$ and $\widehat{\mathfrak{m}_2}$, then $\{\mathfrak{m}_1, \mathfrak{m}_2\}$, as a set of observations, can be simulated using their respective arithmetical expressions and the input-shares $\widehat{\mathfrak{m}_1}$ and $\widehat{\mathfrak{m}_2}$. Hence, if our set of observations is s.t. $|\widehat{O}| < t_1 + t_2$, we can simulate O using at most $t_1 + t_2 - 1$ input shares. Otherwise, since $|O| \leq t_1 + t_2$, it must be that we have exactly $t_1 + t_2$ observations, that we do not observe any random, and that for any two distinct observations ${\mathfrak{m}_\mathbf{1}}^{k_1}$ and ${\mathfrak{m}_\mathbf{2}}^{k_2}$ of O, we have $k_1 \neq k_2$ — we say that O is injective. Assume that we know an observation $\mathfrak{m}$ that does not depend on any observed input shares and s.t. there exists $k \in \overline{\mathfrak{m}} \setminus \overline{O_\mathfrak{m}}$. Then, we can simulate O by first simulating $\mathfrak{m}$ using the random $\mathbf{r}^k$ and then, like in the previous case, by simulating all the observations $O_\mathfrak{m}$ using their respective arithmetical expressions — this last simulation step only depends on randoms different from $\mathbf{r}^k$ and, as we have just seen, uses at most $t_1 + t_2 - 1$ input shares. We conclude the proof by showing that there always exists such an observation $\mathfrak{m}$. Let $\mathfrak{m}^k$ be any non-input share s.t. no non-input observation of form the $\mathfrak{n}^{k+1}$ belongs to O — such an observation always exists by cardinality contraint on O. We have that $k \in \overline{\mathbf{m}^k}$ and $\mathbf{m}^k$ may at most depend on the input share $\mathbf{a}^k$. For k to occur in $\overline{O_{\mathbf{m}^k}}$, it must be that a non-input share of the form $\mathfrak{n}^k$ occurs in $O_{\mathfrak{m}^k}$ or that $\mathbf{c}^{k+1} \in O$. However, both condition are unsatisfied: the first one by injectivity of O, the second one by definition of $\mathbf{m}^k$. Last, again by injectivity of O, we have that $\mathbf{a}^k \notin O$, hence $\mathbf{m}^k$ do not depend on any input share of O. □

Remark 3 *The bound t_1+t_2-1 is reached with the following kind of observations, and is therefore tight:* $\{\mathbf{b}^t, \mathbf{c}^0, \ldots, \mathbf{c}^k, \mathbf{r}^k\}$.

3.2 *t*-SNI Mask Refreshing by Iterating RefreshBlock.

We now aim to show that strong refresh gadgets can be obtained by iterating RefreshBlock gadgets. We prove that for all t, repeating RefreshBlock $\lceil t/3 \rceil$ times

constructs a t-SNI mask refreshing gadget. This coincides with the intuition obtained from the low-order observations made in previous work [5]. We note that Coron [9] had also proposed to iterate a similar additive mask refreshing gadget t times in order to support secure composition. This result is a strict refinement of his.

Proposition 4 *For $x \geq 1$ Gadget $\mathsf{RefreshBlock}^x$ is f^x_{RB}-NI where $f^x_{\mathsf{RB}} : [0, \ldots, t] \longrightarrow [0, \ldots, t]$:*

$$(t_1, t_2) \longmapsto \begin{cases} t_1 & \text{if } t_1 \leq 2x-1 \quad \text{or} \quad t_2 = 0, \\ t_1 + t_2 - x & \text{otherwise.} \end{cases}$$

Proof (Proposition 4). We first note that f_{RB} is such that $f_{\mathsf{RB}}(t_1, t_2) \leq t_1 + t_2$. The proof is by induction on x. For $x = 1$, we note that $f^x_{\mathsf{RB}}(t_1, t_2) = f_{\mathsf{RB}}(t_1, t_2)$ for all t_1, t_2 and conclude by Prop. 3. Assume now that the property holds for x. We prove that it also holds for $x + 1$. We consider the composition:

$$\mathsf{RefreshBlock}^{x+1} = \mathsf{RefreshBlock}^x; \mathsf{RefreshBlock}.$$

We consider a set Ω of observations in $\mathsf{RefreshBlock}^{x+1}$ such that $|\Omega| \leq t$, and partition it into sets of t_x observations internal to $\mathsf{RefreshBlock}^x$, t_1 observations in $\mathsf{RefreshBlock}$, and t_2 output observations. We therefore need to show that any such Ω can be simulated using at most $f^{x+1}_{\mathsf{RB}}(t_x + t_1, t_2)$ shares of its input. By Propositions 1 and 3, it is sufficient to prove $f^x_{\mathsf{RB}}(t_x, f_{\mathsf{RB}}(t_1, t_2)) \leq f^{x+1}_{\mathsf{RB}}(t_x + t_1, t_2)$. Let us first consider the case $t_2 = 0$. In this case, we have:

$$\begin{aligned} f^x_{\mathsf{RB}}(t_x, f_{\mathsf{RB}}(t_1, t_2)) &= f^x_{\mathsf{RB}}(t_x, t_1) \\ &\leq t_x + t_1 = f^{x+1}_{\mathsf{RB}}(t_x + t_1, t_2) \end{aligned}$$

We now have $t_2 \geq 1$.

If $t_1 \leq 1$, we have $f^x_{\mathsf{RB}}(t_x, f(t_1, t_2)) = f^x_{\mathsf{RB}}(t_x, t_1)$.

- If $t_x \leq 2x - 1$, then $f^x_{\mathsf{RB}}(t_x, t_1) = t_x$ and $t_x + t_1 \leq 2x + 1$ since $t_1 \leq 1$. So $f^{x+1}_{\mathsf{RB}}(t_x + t_1, t_2) = t_x + t_1$ and we can conclude;
- Else $2x \leq t_x$. Therefore, we have $f^x_{\mathsf{RB}}(t_x, t_1) = t_x + t_1 - x$ and $f^{x+1}_{\mathsf{RB}}(t_x + t_1, t_2)$ is either $t_x + t_1$ or $t_x + t_1 + t_2 - (x + 1)$ (depending on the value of $t_x + t_1$). The conclusion is trivial in the first case, and follows from $t_2 \geq 1$ in the second case.

We now have $t_2 \geq 1$ and $t_1 \geq 2$. So $f^x_{\mathsf{RB}}(t_x, f_{\mathsf{RB}}(t_1, t_2)) = f^x_{\mathsf{RB}}(t_x, t_1 + t_2 - 1)$.

- If $t_x + t_1 \leq 2x + 1$, and since $t_1 \geq 2$, we have $f^{x+1}_{\mathsf{RB}}(t_x + t_1, t_2) = t_x + t_1$ and $t_x \leq 2x - 1$. So

$$f^x_{\mathsf{RB}}(t_x, t_1 + t_2 - 1) = t_x \leq f^{x+1}_{\mathsf{RB}}(t_x + t_1, t_2)$$

- Else, we have $t_x + t_1 > 2x + 1$ and $t_x > 2x - 1$ since $t_1 \geq 2$. Therefore we have

$$f^x_{\mathsf{RB}}(t_x, t_1 + t_2 - 1) = t_x + t_1 + t_2 - 1 - x = f^{x+1}_{\mathsf{RB}}(t_x + t_1, t_2),$$

which concludes the proof. □

Corollary 2 *RefreshBlock$^{\lceil t/3 \rceil}$ is t-SNI.*

Proof (Corollary 2). We prove that RefreshBlock^x is t-SNI if $x \geq \lceil t/3 \rceil$. By Remark 2 and Proposition 4 it is sufficient to prove $f^x_{\mathsf{RB}}(t_1, t_2) \leq t_1$. If $t_2 = 0$, then $f^x_{\mathsf{RB}}(t_1, t_2) = t_1$ be definition of f^x_{RB} and we can conclude. If $t_1 \leq 2x - 1$ then $f^x_{\mathsf{RB}}(t_1, t_2) = t_1$ by definition of f^x_{RB} and we can conclude. Else, we have $f^x_{\mathsf{RB}}(t_1, t_2) = t_1 + t_2 - x$, and it is thus sufficient to prove $t_2 \leq x$. In this case, we have $t_1 + t_2 \leq t$ (by global constraint), $t_1 \geq 2x$ (since we are not in the case where $t_1 \leq 2x - 1$) and $x \geq \lceil t/3 \rceil$ (by hypothesis), and we conclude that:

$$t_2 \leq t - t_1 \leq t - 2x \leq x.$$

Table 1 summarizes the concrete results Corollary 2 yields for some low orders. To the best of our knowledge, these are currently the best known *parallel* SNI mask refreshing algorithms. We further improve them next.

Order t	2	3	4	5	6	7	8	9	10	11	12
# RefreshBlock	1	1	2	2	2	3	3	3	4	4	4
# randoms	3	4	10	12	14	24	27	30	44	48	52

Table 1: Number of required instances of RefreshBlock and random values for several small orders

4 Optimized Parallel Mask Refreshing

The previous proof followed the identification by Barthe et al. [5] of a pattern leading to low-order secure mask refreshing gadgets. A more systematic exploration of the design space may therefore further reduce the randomness complexity of such gadgets. In this section, we highlight two particular observations that may be of independent interest in other search efforts, and yield particularly interesting performance gains.

4.1 Avoiding Repeated Pairs

A first interesting pattern we identified is that the most efficient mask refreshing gadgets at low orders rarely involve repeated pairs of random variables (i.e., they rarely include both the pattern $a_i \oplus r_j \oplus r_k$ and $a_j \oplus r_j \oplus r_k$ for $i \neq j$). However, looking at the definition of our RefreshBlock$^{\lceil t/3 \rceil}$ mask refreshing algorithm in Section 3.2, we observe that the same pairs of shares are involved in the same way in all iterations of RefreshBlock.

We thus consider a slight variant RefreshBlock$_j$ of RefreshBlock, shown as Algorithm 2, that rotates its vector of randomness by j instead of 1 only. By

composing $\mathsf{RefreshBlock}$ with $\mathsf{RefreshBlock}_j$ for some well-chosen j, we therefore avoid the repetition of patterns in the use of shares and randomness during successive iterations of the algorithm.

Alg. 2 Rotation-Parameterized $\mathsf{RefreshBlock}_j$.

```
function RefreshBlock_j(a)
    for i = 0 to t do
        r^i <-$ K
    for i = 0 to t do
        a^i <- a^i ⊕ r^i
        a^i <- a^i ⊕ r^{i-j}
    return a
```

In this exploration experiment, we look for sequences of rotation offsets that allow us to reduce the randomness and time complexity of the mask refreshing gadget whilst still achieving t-SNI security.

Table 2 summarizes the results of our exploration for 8, 9, 10, 11 and 12 shares. For all, we find solutions that require only two iterations of $\mathsf{RefreshBlock}_j$ (with different values of j). This improves on the general result obtained by applying Corollary 2, but preserves the parallel nature of the mask refreshing algorithm.

Our exploration in this setting further fails to find any SNI algorithms with only 2 iterations of $\mathsf{RefreshBlock}_j$ for 12 shares, although it notably finds an algorithm for 12 shares with 3 iterations of $\mathsf{RefreshBlock}_j$, i.e.,

$$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_2; \mathsf{RefreshBlock}_3$$

with a cost of 36 random field elements.

We note some interesting symmetries in the results due to the fact that a rotation of i positions in one direction is equivalent to a rotation of $t+1-i$ positions in the other. Interestingly, it also appears that, at order t, the gadget $\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_{(t+1)/2}$ is never t-SNI when $t+1$ is even.

4.2 Breaking Chains

Out of the results of the first wave of parallel synthesis, we make a second observation: attacks found by the verification tool in the failure cases rely on the construction of "chains" of observations to break SNI. More specifically, an adversary that makes a particular observation, in order to propagate it back to input shares, needs to carefully select intermediate observations that "lock" the random variables used as masks on that share, and further expend an internal observation to "cap off" the chain, blocking the final random variable in it. For example, considering $\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_2$ for $t = 10$, the observations marked below (in brackets $[\cdot]$) are a counter-example to SNI: they include 4 output observation and 6 internal observations whose distribution depends on 7 input shares.

Algorithm	# rand.	SNI?
$t = 7$		
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_1$	16	✗
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_2$	16	✓
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_3$	16	✓
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_4$	16	✗
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_5$	16	✓
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_6$	16	✓
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_7$	16	✗
$t = 8$		
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_1$	18	✗
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_2$	18	✓
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_3$	18	✓
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_4$	18	✓
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_5$	18	✓
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_6$	18	✓
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_7$	18	✓
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_8$	18	✗
$t = 9$		
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_1$	20	✗
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_2$	20	✓
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_3$	20	✓
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_4$	20	✓
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_5$	20	✗
$t = 10$		
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_1$	22	✗
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_2$	22	✗
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_3$	22	✓
$\mathsf{RefreshBlock}_1; \mathsf{RefreshBlock}_4$	22	✓

Table 2: Some parallel mask refreshing gadgets.

$$
\begin{array}{lllllllll}
\mathbf{c}^0 = [\mathbf{a}^0 & \oplus & [\mathbf{r_1}^0] & \oplus & \mathbf{r_1}^{10} & \oplus & [\mathbf{r_2}^0] & \oplus & \mathbf{r_2}^9] \\
\mathbf{c}^1 = [\mathbf{a}^1 & \oplus & \mathbf{r_1}^1 & \oplus & \mathbf{r_1}^0 & \oplus & [\mathbf{r_2}^1] & \oplus & \mathbf{r_2}^{10}] \\
\mathbf{c}^2 = \mathbf{a}^2 & \oplus & \mathbf{r_1}^2 & \oplus & \mathbf{r_1}^1 & \oplus & \mathbf{r_2}^2 & \oplus & \mathbf{r_2}^0 \\
\mathbf{c}^3 = \mathbf{a}^3 & \oplus & \mathbf{r_1}^3 & \oplus & \mathbf{r_1}^2 & \oplus & \mathbf{r_2}^3 & \oplus & \mathbf{r_2}^1 \\
\mathbf{c}^4 = \mathbf{a}^4 & \oplus & \mathbf{r_1}^4 & \oplus & \mathbf{r_1}^3 & \oplus & \mathbf{r_2}^4 & \oplus & \mathbf{r_2}^2
\end{array}
$$

$$
\begin{array}{llllllll}
\mathbf{c}^5 = \mathbf{a}^5 & \oplus & \mathbf{r_1}^5 & \oplus & \mathbf{r_1}^4 & \oplus & \mathbf{r_2}^5 & \oplus & \mathbf{r_2}^3 \\
\mathbf{c}^6 = [\mathbf{a}^6 & \oplus & \mathbf{r_1}^6] & \oplus & \mathbf{r_1}^5 & \oplus & \mathbf{r_2}^6 & \oplus & \mathbf{r_2}^4 \\
\mathbf{c}^7 = [\mathbf{a}^7 & \oplus & \mathbf{r_1}^7 & \oplus & \mathbf{r_1}^6 & \oplus & \mathbf{r_2}^7] & \oplus & \mathbf{r_2}^5 \\
\mathbf{c}^8 = [\mathbf{a}^8 & \oplus & \mathbf{r_1}^8 & \oplus & \mathbf{r_1}^7 & \oplus & \mathbf{r_2}^8] & \oplus & \mathbf{r_2}^6 \\
\mathbf{c}^9 = [\mathbf{a}^9 & \oplus & \mathbf{r_1}^9 & \oplus & \mathbf{r_1}^8 & \oplus & \mathbf{r_2}^9 & \oplus & \mathbf{r_2}^7] \\
\mathbf{c}^{10} = [\mathbf{a}^{10} & \oplus & \mathbf{r_1}^{10} & \oplus & \mathbf{r_1}^9 & \oplus & \mathbf{r_2}^{10} & \oplus & \mathbf{r_2}^8]
\end{array}
$$

A key observation is that those intermediate observations that serve to construct the chain ($\mathbf{a}^8 \oplus \mathbf{r_1}^8 \oplus \mathbf{r_1}^7 \oplus \mathbf{r_2}^8$, $\mathbf{a}^7 \oplus \mathbf{r_1}^7 \oplus \mathbf{r_1}^6 \oplus \mathbf{r_2}^7$ and $\mathbf{a}^6 \oplus \mathbf{r_1}^6$) do not simply lock random variables to make them unusable by the simulator: they also let the adversary gain information about one additional input share, leading to the attack. In short: those intermediate observations give more information to the adversary by letting her observe an input-dependent distribution *and* remove power from the simulator by locking random variables away.

With this observation in mind, it seems natural to consider a slight twist on the construction: instead of immediately adding the sampled randomness onto the input, we could construct an encoding of 0 using the same rotation-based technique and only once enough randomness has been mixed in add it onto the input encoding $\mathbf{a}$. The consequence of this modification (which we generalize below) on the attack above is that intermediate observations can remove power from the simulator by locking random variables, but will never give any information about input shares to the adversary.

We now define general algorithms embodying this approach to mask refreshing, and state some results. Consider Algorithm 3a, which corresponds to the input-free part of $\mathsf{RefreshBlock}_j$ that computes an encoding of 0. We denote with $\mathsf{RefreshZero}^t_\ell(\mathbf{a})$ (with ℓ a list of rotation offsets for use in $\mathsf{ZeroBlock}$) the algorithm that produces an initial encoding of 0 using $\mathsf{ZeroBlock}$ (with the appropriate rotation offset) and remasks it using $\mathsf{RefreshBlock}$ (with the appropriate rotation offset) before adding the final value onto the input shares.[9]

[9] We note that another algorithm can be obtained by producing multiple encodings of 0 via $\mathsf{ZeroBlock}$ and adding them together after they are produced. This algorithm has a marginally higher memory complexity, and we believe this makes no difference to security – which we leave as a scope for further investigations.

Alg. 3 The $\mathsf{ZeroBlock}_j$ algorithm and its usage.

function $\mathsf{ZeroBlock}_j^t(())$
  for $i = 0$ **to** t **do**
    $\mathbf{r}^i \xleftarrow{\$} \mathbb{K}$
  for $i = 0$ **to** t **do**
    $\mathbf{c}^i \leftarrow \mathbf{r}^i$
    $\mathbf{c}^i \leftarrow \mathbf{c}^i \oplus \mathbf{r}^{i-j}$
  return c

(3a) $\mathsf{ZeroBlock}_j$.

function $\mathsf{RefreshZero}_{[1;2]}^{10}((\mathbf{a}))$
  $\mathbf{r} \leftarrow \mathsf{ZeroBlock}_1^{10}()$
  $\mathbf{r} \leftarrow \mathsf{RefreshBlock}_2^{10}(\mathbf{r})$
  $\mathbf{c} \leftarrow \mathbf{r} \oplus \mathbf{a}$

(3b) A modified parallel mask refreshing algorithm based on $\mathsf{ZeroBlock}_j^t$ for $t = 10$.

For example, for $t = 10$, the modified algorithm described above is defined as $\mathsf{RefreshZero}_{[1;2]}^{10}$, shown as Algorithm 3b. Using Barthe et al.'s maskVerif tool [4], we obtain the results shown in Table 3, demonstrating the existence of a parallel t-SNI mask refreshing algorithm that uses only $2(t+1)$ random elements in $\mathbb{K}$ for masking orders up to $t = 16$. In particular, this range includes $t = 7$ and $t = 15$, which are particularly useful for masking bitsliced implementations. Due to the complexity of the verification problems, we do not know the minimal order for which a third encoding of 0 is required. Still, it is worth noting that, although our most effective improvements are limited to relatively low orders, they can be used to yield significant improvements also at very high orders. Indeed, if used in combination with Battistello et al's recursive mask refreshing algorithm, our optimized low-order gadgets improve the state-of-the-art in randomness complexity for mask refreshing as shown in the last column of Table 4 (where a † is used to signify that the RefreshZero algorithm is used as a base case in the improved algorithm, and a − is used to denote the fact that Battistelo et al's algorithm remains the best known at this order). We note in particular the significant reduction in randomness complexity for mask refreshing at order $t = 2^n - 1$.

A note on horizontal attacks. We conclude by observing that the RefreshZero algorithm was suggested (without proof) for another purpose in [21]. Indeed, another advantage of this algorithm (which cannot be captured in the probing model) is that it increases the resistance to horizontal attacks, whereby an adversary may combine several time points from the leakage traces to learn more information about a sensitive variable. Since the sensitive variable is used only once in RefreshZero, it reduces the impact of such attacks at gadget level.

Acknowledgments. François-Xavier Standaert is a senior research associate of the Belgian Fund for Scientific Research (F.R.S.-FNRS). This work has been funded in parts by the European Union through the ERC project SWORD (724725).

Algorithm	SNI	Verif. OoM
$\mathsf{RefreshZero}^{2}_{[1]}$	✓	ϵ
$\mathsf{RefreshZero}^{3}_{[1]}$	✓	ϵ
$\mathsf{RefreshZero}^{4}_{[1]}$	✓	ϵ
$\mathsf{RefreshZero}^{5}_{[1]}$	✗	-
$\mathsf{RefreshZero}^{5}_{[2]}$	✗	-
$\mathsf{RefreshZero}^{5}_{[1,2]}$	✓	ϵ
$\mathsf{RefreshZero}^{6}_{[1,2]}$	✓	ϵ
$\mathsf{RefreshZero}^{7}_{[1,1]}$	✓	ϵ
$\mathsf{RefreshZero}^{7}_{[1,2]}$	✓	ϵ
$\mathsf{RefreshZero}^{8}_{[1,1]}$	✓	ϵ
$\mathsf{RefreshZero}^{8}_{[1,2]}$	✓	ϵ
$\mathsf{RefreshZero}^{9}_{[1,1]}$	✗	-
$\mathsf{RefreshZero}^{9}_{[1,2]}$	✓	2s
$\mathsf{RefreshZero}^{9}_{[1,1,1]}$	✓	3s
$\mathsf{RefreshZero}^{10}_{[1,2]}$	✓	18s
$\mathsf{RefreshZero}^{11}_{[1,2]}$	✓	49s
$\mathsf{RefreshZero}^{12}_{[1,2]}$	✓	10min
$\mathsf{RefreshZero}^{13}_{[1,2]}$	✗	-
$\mathsf{RefreshZero}^{13}_{[1,3]}$	✓	30min
$\mathsf{RefreshZero}^{14}_{[1,3]}$	✓	1.5h
$\mathsf{RefreshZero}^{15}_{[1,3]}$	✓	2h
$\mathsf{RefreshZero}^{16}_{[1,3]}$	✓	2d
$\mathsf{RefreshZero}^{17}_{[1,3]}$	✗	2h
$\mathsf{RefreshZero}^{17}_{[1,4]}$	✓	3d
$\mathsf{RefreshZero}^{18}_{[1,4]}$	✓	1 month

Table 3: Order of Magnitude (OoM) of time taken for the verification of some RefreshZero-based mask refreshing gadgets.

order t	[6]	RB	RZ	[6] + RZ
1	1	2 ([5])	2	-
2	3	3 ([5])	3	†
3	6	4 ([5])	4	†
4	8	10 ([5])	10	-
5	12	12 ([5])	12	†
6	15	14 ([5])	14	13
7	20	24 ([5])	16	†
8	22	27 ([5])	18	†
9	26	30 ([5])	20	†
10	30	44	22	†
11	36	48	24	†
12	39	52	26	†
13	44	70	28	†
14	49	75	30	†
15	56	80	32	†
16	58	102	34	†
17	62	108	36	†
18	66	114	38	†
19	72	120	✗	60
20	76	126	✗	62
		...		
31	144	320	✗	96
63	352	1344	✗	256

Table 4: Best known mask refreshing for some orders.

References

1. Martin R. Albrecht and Kenneth G. Paterson. Lucky microseconds: A timing attack on amazon's s2n implementation of TLS. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 622–643. Springer, Heidelberg, May 2016.
2. Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, and Benjamin Grégoire. Compositional verification of higher-order masking: Application to a verifying masking compiler. Cryptology ePrint Archive, Report 2015/506, 2015. `http://eprint.iacr.org/2015/506`.
3. Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, and Pierre-Yves Strub. Verified proofs of higher-order masking. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 457–485. Springer, Heidelberg, April 2015.
4. Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, Benjamin Grégoire, Pierre-Yves Strub, and Rébecca Zucchini. Strong non-interference and type-directed higher-order masking. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 116–129. ACM Press, October 2016.
5. Gilles Barthe, François Dupressoir, Sebastian Faust, Benjamin Grégoire, François-Xavier Standaert, and Pierre-Yves Strub. Parallel implementations of masking schemes and the bounded moment leakage model. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 535–566. Springer, Heidelberg, May 2017.
6. Alberto Battistello, Jean-Sébastien Coron, Emmanuel Prouff, and Rina Zeitoun. Horizontal side-channel attacks and countermeasures on the ISW masking scheme. In Benedikt Gierlichs and Axel Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 23–39. Springer, Heidelberg, August 2016.
7. Sonia Belaïd, Fabrice Benhamouda, Alain Passelègue, Emmanuel Prouff, Adrian Thillard, and Damien Vergnaud. Randomness complexity of private circuits for multiplication. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 616–648. Springer, Heidelberg, May 2016.
8. Claude Carlet, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Algebraic decomposition for probing security. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 742–763. Springer, Heidelberg, August 2015.
9. Jean-Sébastien Coron. Higher order masking of look-up tables. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 441–458. Springer, Heidelberg, May 2014.
10. Jean-Sébastien Coron, Emmanuel Prouff, and Matthieu Rivain. Side channel cryptanalysis of a higher order masking scheme. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES 2007*, volume 4727 of *LNCS*, pages 28–44. Springer, Heidelberg, September 2007.
11. Jean-Sébastien Coron, Emmanuel Prouff, Matthieu Rivain, and Thomas Roche. Higher-order side channel security and mask refreshing. In Shiho Moriai, editor, *FSE 2013*, volume 8424 of *LNCS*, pages 410–424. Springer, Heidelberg, March 2014.
12. Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 423–440. Springer, Heidelberg, May 2014.

13. Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 401–429. Springer, Heidelberg, April 2015.
14. Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 135–156. Springer, Heidelberg, May 2010.
15. Wieland Fischer and Naofumi Homma, editors. *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*. Springer, 2017.
16. Daniel Genkin, Lev Pachmanov, Itamar Pipman, Eran Tromer, and Yuval Yarom. ECDSA key extraction from mobile devices via nonintrusive physical side channels. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 16*, pages 1626–1638. ACM Press, October 2016.
17. Dahmun Goudarzi and Matthieu Rivain. How fast can higher-order masking be in software? In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 567–597. Springer, Heidelberg, May 2017.
18. Hannes Groß and Stefan Mangard. Reconciling d+1 masking in hardware and software. In Fischer and Homma [15], pages 115–136.
19. Daniel Gruss, Raphael Spreitzer, and Stefan Mangard. Cache template attacks: Automating attacks on inclusive last-level caches. In *24th USENIX Security Symposium, USENIX Security 15, Washington, D.C., USA, August 12-14, 2015.*, pages 897–912, 2015.
20. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481. Springer, Heidelberg, August 2003.
21. Anthony Journault and François-Xavier Standaert. Very high order masking: Efficient implementation and security evaluation. In Fischer and Homma [15], pages 623–643.
22. Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 388–397. Springer, Heidelberg, August 1999.
23. Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 142–159. Springer, Heidelberg, May 2013.
24. Matthieu Rivain and Emmanuel Prouff. Provably secure higher-order masking of AES. In Stefan Mangard and François-Xavier Standaert, editors, *CHES 2010*, volume 6225 of *LNCS*, pages 413–427. Springer, Heidelberg, August 2010.
25. Kai Schramm and Christof Paar. Higher order masking of the AES. In David Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 208–225. Springer, Heidelberg, February 2006.
26. Eran Tromer, Dag Arne Osvik, and Adi Shamir. Efficient cache attacks on AES, and countermeasures. *Journal of Cryptology*, 23(1):62–74, January 2010.